TRIUMPH ADLER

TA Computer Systeme

TA 1600

TA 1630

TAXO

SYSTEM COMMAND INTERPRETER

30.06.80

Titel : TA 1630 TAXO System Command Interpreter 26.06.80

Druck-Nummer : TA1630/SCI/0780/d

EINE BITTE AN DIE BENUTZER

Wenn Sie diese Empfangsbestätigung ausgefüllt an uns zurücksenden, werden wir Ihnen, soweit dies notwendig ist, gern Ergänzungen oder Korrekturmitteilungen zukommen lassen.

EMPFANGSBESTÄTIGUNG

Der Empfänger versichert, diese Unterlagen nur bestimmungsgemäß zu verwenden und sie unbefugten Dritten nicht zu überlassen, keine Vervielfältigungen oder Nachdrucke (auch auszugsweise) anzufertigen oder sonst ihren Inhalt ganz oder teilweise anderen Personen zugänglich zu machen. Der Empfänger wird seine Mitarbeiter entsprechend verpflichten.

Alle Rechte, insbesondere Urheberrechte, sind TAV vorbehalten.

Name, Vorname .....

Firma .....

Abteilung .....

Ort .....

Straße .....

Datum, Unterschrift .....

Bitte an umseitige Adresse schicken!

Triumph-Adler
Vertriebs-GmbH.
Abt. VMV 4
Fürther Str. 212
8500 Nürnberg

INHALTSVERZEICHNIS

1 DIE VERWENDUNG DES SYSTEM COMMAND INTERPRETERS
1.1 Allgemeines
1.2 Zweck des SCI
1.3 Anwendung des SCI
1.4 Begriffsbestimmung

2 SCI ÜBERBLICK
2.1 Kommandoprozedur
2.2 Kommandoprozessor
2.3 Installierung von Kommandoprozeduren und Prozessoren
2.4 Verwendung der Kommandos
2.5 Löschen eines Kommandos

3 ENTWURF VON KOMMANDOPROZEDUREN UND PROZESSOREN
3.1 Entwurf von Prozeduren
3.2 Entwurf von Kommando Prozeduren

4 DIE SYSTEM COMMAND INTERPRETER SPRACHE
4.1 Primitiv Kommando für die Prozedur-Definition
4.2 Primitiv Kommandos für die Synonymbehandlung
4.3 Das Primitiv Kommando DATA
4.4 Primitiv Kommandos zum Aufruf von Kommando Prozessoren
4.5 Bedingte Primitiv Kommandos
4.6 Verlasse eine Prozedur
4.7 Verwendung einer alternativen Prozedur Bibliothek
4.8 Zeige eine Datei
4.9 Gib eine Meldung aus
4.10 Beende die Arbeit des SCI
4.11 Zeige ein Menü
4.12 Iterative Schleifen

5 MENÜS

1 DIE VERWENDUNG DES SYSTEM COMMAND INTERPRETERS

1.1 Allgemeines

Der System Command Interpreter (SCI) ist eine einheitliche Schnittstelle zwischen dem Anwender und dem Betriebssystem TAXO, den Systemdiensten, dem Softwareentwicklungssystem und den Anwenderprogrammen. Jedes beliebige Anwenderprogramm kann als Schnittstelle zwischen dem Anwender und dem Bediener dienen, indem man anwendungsorientierte SCI-Kommandos und Menüs definiert.

Dieses Kapitel beschreibt, wie der SCI arbeitet und wie er dazu verwendet werden kann, neue Kommandos und neue Menüs zu erzeugen.

1.2 Zweck des SCI

Der System Command Interpreter kann dazu verwendet werden, Programme, die unter dem Betriebssystem laufen, aufzurufen. Außerdem kann der Anwender diesen Programmen über den SCI Laufzeitparameter mitgeben. Der SCI erlaubt jedem Anwender Tabellen von Variablen, genannt Synonyme, aufzubauen, zu verwalten und diesen Variablen Werte zuzuweisen. Mit Hilfe des SCI können Anwendungsprogramme auf diese Synonyme zugreifen. Ebenso können die Anwendungsprogramme über der SCI dazu verwendet werden, Folgen von Kommandos im Stapel abzuarbeiten (Batch-Verarbeitung).

ANMERKUNG

Der SCI ist nicht als Datenerfassungssystem gedacht, oder um große Mengen von Daten (z.B. Text) an ein Programm zu übergeben.

1.3 Anwendung des SCI

Um mit Hilfe des SCI ein Anwenderprogramm aufzurufen, muß der Anwender ein SCI Kommando definieren, welches sein Programm startet. Die vom Anwender definierten Kommandos können in das Repertoire der schon vorhandenen SCI Kommandos eingefügt werden, oder können in einer eigenen Kommando Bibliothek zusammengefaßt werden. Unter Verwendung des Primitiv-Kommandos .USE (beschrieben im Paragraph 4.7) kann der Anwender angeben, welche Kommandobibliothek von SCI verwendet werden soll.

Für Systeme, die normalerweise nur Anwenderprogramme und die zugehörigen Kommandos verwenden, kann der Anwender die originale SCI-Kommandobibliothek in eine andere Bibliothek kopieren, während er die Anwendungsbibliothek an die Stelle der SCI Bibliothek schreibt. Für solche Systeme ist es zweckmäßig die normalen Menüs, die unter dem SCI verfügbar sind, zu ersetzen. Im Paragraph 5 wird das Ändern und Neuschreiben von Menüs erläutert.

1.4 Begriffsbestimmung

Die folgenden Absätze sollen den Anwender mit den verschiedenen Konzepten und Begriffen vertraut machen, die dem System Command Interpreter zuzuordnen sind.

Die SCI Sprache besteht aus allen Kommandos, Primitivkommandos und Menüs, die dem SCI eingegeben werden können und von ihm interpretiert werden (z.B. als Antwort auf die Kommandoanfrage [ ]). Beispiele für Kommandos sind: IT, AS, XE. Die Primitivkommandos sind in Paragraph 4 beschrieben. Menünamen sind 1-6 ASCII Zeichen lang. Ein Menü wird angezeigt, wenn <Menüname> als Kommando eingegeben wird. Jedes einzelne Kommando, Primitivkommando oder jeder Menüaufruf ist ein SCI Kommando.

Eine Kommandoprozedur ist eine Folge von Anweisungen, die in der SCI Sprache geschrieben ist und ein Kommando für den SCI definiert. Immer wenn ein Kommando (nicht ein Primitivkommando) dem System Command Interpreter übergeben wird, sucht der SCI die betreffende Kommandoprozedur in der Bibliothek für diese Prozeduren und führt diese Prozedur aus, indem er eine Anweisung nach der anderen interpretiert.

Eine Prozedurbibliothek besteht aus einem Directory, das für jede Kommandoprozedur eine sequentielle Datei enthält. Die SCI Systembibliothek heißt .S$PROC. Es ist zu beachten, daß die oben beschriebene Methode der Ausführung von Kommandos die SCI Sprache zu einer Interpretersprache macht. Eine Ausnahme von dieser Methode der Ausführung gilt für die Primitivkommandos, die im System Command Interpreter selbst interpretiert werden.

Ein Kommandoprozessor ist eine Task, welche unter dem Betriebssystem TAXO arbeitet und von einer Kommandoprozedur aufgerufen wird, um die von dem betreffenden Kommando geforderte Aktion durchzuführen. Kommandoprozessoren müssen nicht eindeutig einer Kommandoprozedur zugeordnet sein. Eine Prozedur kann mehrere Prozessoren aufrufen oder ein Prozessor kann von mehreren verschiedenen Prozeduren verwendet werden.

Ein Menü ist eine sequentielle Datei mit einer Satzlänge von 80 Zeichen und maximal 23 Sätzen. Menüs sollen dazu dienen, dem SCI Anwender das Auffinden desjenigen Kommandos zu erleichtern, dessen Funktion er ausführen will. Daher soll ein Menü die Namen von anderen Menüs und/ oder Kommandos enthalten. Menüs werden detaillierter in Paragraph 4 und 5 beschrieben.

Synonyme sind Variable der SCI Sprache. Sie können als Wert entweder Null oder eine Kette von Zeichen besitzen. Die leere Zeichenkette ist einfach eine Zeichenkette der Länge Null, und wird durch zwei unmittelbar aufeinanderfolgende Doppelanführungszeichen (") dargestellt. Die maximale Länge eines Synonymnamens oder Wertes beträgt 255 Zeichen.

Ein anderer Typ von Variable in der SCI Sprache ist das Schlüsselwort. Schlüsselworte sind Variable, die auf dem Bildschirm angezeigt werden, wenn ein Kommando interaktiv an den SCI übergeben wird (z.B. für das Kommando SF (Show File), ist FILE PATHNAME das einzige Schlüsselwort). Die Werte, die durch den Anwender eingegeben werden, werden der Schlüsselwortvariablen zugeordnet und stehen so der Kommandoprozedur zur Verfügung. Die verschiedenen Typen von Schlüsselwortwerten sind in Paragraph 4 beschrieben.

Die Nachrichtendatei ist eine sequentielle Datei (.S$NEWS), die vom SCI unmittelbar nachdem sich der Bediener beim System angemeldet hat (Log-in) auf dem Bildschirm angezeigt wird. Die Nachrichtendatei kann dazu verwendet werden, die Benutzer des Systems über Änderungen oder Probleme mit dem System zu informieren, oder beliebige andere Informationen, welche von einem allgemeinen Interesse sind, an alle Systembenutzer auszugeben. Die Nachrichtendatei kann einfach mit Hilfe des Texteditors verändert werden.

Die Terminal Communication Area (TCA) ist ein Speicherbereich innerhalb des SCI, welche die Kommunikation zwischen Datensichtgerät und dem System unterstützt. Bei jeder Aktivierung des SCI an einem Terminal wird eine TCA aufgebaut. Die TCA erfüllt drei Aufgaben: sie enthält die Beschreibung des aktuellen Benutzers; sie enthält die Tabelle der Synonyme und ihrer Werte (name correspondence table); und schließlich dient die TCA dazu Informationen zwischen SCI und den Kommandoprozessoren auszutauschen.

Jedem Terminal können zwei spezielle Tasks, die unter dem SCI arbeiten, zugeordnet sein; eine Task im Vordergrund (foreground task) und eine Task im Hintergrund (background task). Beide Tasks unterscheiden sich von einer Task, die durch das XT-Kommando (Execute Task) aktiviert worden ist, durch den Zugriff auf die TCA (und damit auf die Synonyme und Schlüsselworte und deren Werte). Eine Task im Vordergrund ist durch die Eingabe von einem Terminal und die Ausgabe auf dieses Terminal gekennzeichnet, d.h. eine Task im Vordergrund arbeitet interaktiv mit dem Terminal. Zum Beispiel arbeitet der SCI, der von einem Terminal aus aktiviert wird, für dieses Terminal im Vordergrund. Andere Tasks können mit Hilfe des BID Primitivkommandos im Vordergrund ausgeführt werden.

Eine Task im Hintergrund ist eine Task, welche von einem Terminal mit Hilfe des .QBID Primitivkommandos (siehe Paragraph 4) gestartet wird und nicht interaktiv mit dem Terminal arbeitet. Der Sprachprozessor für COBOL ist ein Beispiel für eine Hintergrund Task. Wenn eine solche Task beendet ist, wird an das Terminal eine Meldung ausgegeben. Der SCI kann durch das XB Kommando (Execute Batch) als Hintergrund Task aufgerufen werden. Wenn der SCI als Hintergrund Task arbeitet, erwartet er die Eingaben im Batchmodus, d.h. er erwartet eine sequentielle Folge von Kommandos im Batchformat (siehe Bedienungshandbuch II)

Wenn eine Hintergrund Task aufgerufen wird, wird dieser Task eine Kopie der TCA zur Verfügung gestellt. Diese Kopie ist vollständig von der TCA des Vordergrundes getrennt und wird nicht dynamisch aktualisiert, wie dies mit der Vordergrund TCA geschieht. Dies bedeutet, daß eine Hintergrund Task keinen Zugriff auf Synonyme besitzt, nachdem die Task gestartet wurde. Ebenso sind die Synonyme, die bei einer Hintergrundverarbeitung zugewiesen sind, im Vordergrund nicht verfügbar.

Von einer Vordergrund oder einer Hintergrund Task können Meldungen an das Terminal gesendet werden, indem diese Meldungen in eine Datei geschrieben werden, die diesem Terminal zugeordnet sind. Diese Datei heißt terminal lokal (terminal local file, TLF). Ein TLF ist eine sequentielle Datei, die nur ASCII Zeichen enthält. Für jedes Terminal, an dem der SCI aktiviert wird, existieren zwei lokale Dateien, eine für den Vordergrund und eine für den Hintergrund. Der TLF wird regelmäßig nach jeder Beendigung eines Kommandos angezeigt. Der TLF für den Hintergrund wird nur dann angezeigt, wenn das SBS-Kommando (Show Background Status) ausgeführt wird und wenn die Hintergrund Task geendet hatte und den TLF beschrieben hatte. Falls der SCI im Batch Modus arbeitet, wird der Hintergrund TLF auf die Listen Datei oder Gerät ausgegeben, wenn das Hintergrund Kommando abgeschlossen ist.

2 SCI ÜBERBLICK

Wie in Paragraph 1.2 erwähnt wird, kann der System Command Interpreter dazu verwendet werden, neue Kommandos für spezielle Anwendungen zu schreiben. Ein neues Kommando zu schreiben, schließt die Erstellung einer Kommando Prozedur ein. Gegebenenfalls muß auch ein Kommandoprozessor geschrieben werden, wenn die Prozedur einen Prozessor aufruft. Sowohl die Prozedur als auch der Prozessor müssen installiert sein, wenn der Anwender das Kommando durch den SCI ausführen lassen will. Die folgenden Paragraphen beschreiben diesen Vorgang unter Verwendung von einfachen Beispielen.

2.1 Kommandoprozedur

Eine Kommandoprozedur ist eine Folge von SCI Anweisungen (Kommandos, Primitivkommandos, Menüaufrufen), welche durch den SCI ausgeführt werden, wenn das Kommando an den SCI übergeben wird. Eine Kommandoprozedur ist ein Programm in einer höheren Programmiersprache, das durch den SCI ausgeführt wird. Das Kommando und die zugeordneten Schlüsselworte (maximal 10 interaktive Schlüsselworte) werden in der Kommandoprozedur durch das Primitivkommando .PROC definiert, das in Paragraph 4.1 im Detail beschrieben ist. Das Primitivkommando .USE gibt an, in welcher Prozedurbibliotzhek das neue Kommando definiert ist.

Das folgende Beispiel zeigt eine einfache Kommandoprozedur, welche zwei Schlüsselworte definiert, EXAMPLE NAME und NUMBER. Diese Prozedur ruft einen Kommandoprozessor auf und übergibt an ihn die beiden Schlüsselworte. Es ist zu beachten, daß Prozeduren nicht unbedingt einen Prozessor aufrufen müssen; sie können ebenso nur andere Prozeduren aufrufen oder einfach Menüs anzeigen.

```
.USE      .USERLIB
.PROC     EXP(EXAMPLE PROCEDURE),
          EXAMPLE NAME= STRING("SAMPLE"),
          NUMBER=INT
        .BID TASK= 55,PARMS=("&EXAMPLE NAME","&NUMBER")
.EOP
```

Nachdem diese Prozedur und der Prozessor, den sie aufruft (Task 55_{16}), richtig installiert sind (siehe Paragraph 2.3), kann sie aufgerufen werden. Nachdem der SCI die Kommandoeingabe anfordert ([ ]), kann die Prozedur mit EXP (Name des Kommandos) aufgerufen werden. Der SCI zeigt dann den vollen Prozedurnamen an (EXAMPLE PROCEDURE) einschließlich der zwei Schlüsselworte. Auf dem Bildschirm des Terminals wird also folgendes Bild angezeigt:

```
[ ]EXP
EXAMPLE PROCEDURE
   EXAMPLE NAME: SAMPLE
      NUMBER:
```

In Paragraph 4 werden die Anweisungen, die in der Beispielprozedur verwendet werden, genau beschrieben.

2.2 Kommandoprozessor

Ein Kommandoprozessor ist eine Task, die von einer Kommandoprozedur aufgerufen wird. Der Prozessor kann ein Anwendungsprogramm sein, das in einer Sprache geschrieben ist, die das Betriebssystem TAXO unterstützt. Der Prozessor kann aber auch ein spezielles Programm sein, das nur ein bestimmtes Kommando ausführt. In jedem Fall werden die Kommandoprozessoren von der Kommandoprozedur mit Hilfe der Primitivkommandos .BID oder.QBID aufgerufen. Das Primitvkommando .BID ruft den Prozessor als Vordergrund Task auf, während .QBID den Prozessor als Hintergrund Task aufruft. Während der Ausführungszeit des Prozessors ist der SCI suspendiert, es sei denn der Prozessor wurde durch .QBID aktiviert. Diese Primitivkommandos werden in Paragraph 4 genau beschrieben.

2.3 Installierung von Kommandoprozeduren und Prozessoren

Bevor ein neues Kommando verwendet werden kann, muß die Kommandoprozedur in der Prozedurbibliothek installiert werden. Falls ein Kommandoprozessor verwendet wird, muß dieser in einer Programmdatei installiert werden.

Kommandoprozeduren können entweder interaktiv am Terminal installiert werden, indem die Prozedur direkt dem SCI eingegeben wird, oder im Batchmodus. In jedem Falle sind die benötigten Anweisungen dieselben. Falls die Prozedur in einer anderen Kommandobibliothek als der Systemkommandobibliothek installiert werden soll, ist für eine solche Bibliothek ein Directory anzulegen. Anschließend ist dem System Command Interpreter mit Hilfe des Primitivkommandos .USE der Pathname der Bibliothek mitzuteilen, in der die Prozedur installiert werden soll:

.USE <Pathname>

Nach der .USE Anweisung werden einfach die Anweisungen der Kommandoprozedur selbst eingegeben. Falls mehr als eine Prozedur in dieselbe Bibliothek einzutragen ist, wird kein weiteres .USE Primitivkommando benötigt.

```
[]CFDIR        PATHNAME = SYSVOL.MYLIB, MAXENT = 101
[].USE         SYSVOL.MYLIB
[].PROC        EXP (EXAMPLE PROCEDURE),
               EXAMPLE NAME = *STRING ("SAMPLE"),
               NUMBER = INT
   .BID          TASK = >55, PARMS = (&EXAMPLE NAME, &NUMBER)
  .EOP
```

[ ] < nächstes Kommando >

Das obige Beispiel zeigt, wie eine Kommandoprozedur interaktiv definiert werden kann. Falls der Anwender mehrere Prozeduren definieren will, oder Tippfehler korrigieren will, ist es besser die zu definierenden Prozeduren mit Hilfe des Texteditors in eine Datei zu schreiben und dann diese Datei im Batchmodus zu verarbeiten (unter Verwendung des XB Kommandos). In diesem Fall ist die obige Kommandoprozedur wie folgt zu installieren:

1. Verwende das XE Kommando ohne Angabe für FILE ACCESS NAME.
2. Versetze den Texteditor in den Composemodus und editiere die obigen SCI Anweisungen in die Schmierdatei.
3. Verwende das QE Kommando und antworte bei ABORT mit NO. Für den OUTPUT FILE ACCESS NAME ist eine temporäre Datei (z.B. .PROC) anzugeben.
4. Verwende das XB Kommando, wobei die temporäre Datei bei INPUT ACCESS NAME anzugeben ist. Die Anfrage für LISTING ACCESS NAME kann beliebig beantwortet werden.

Wenn die Batchverarbeitung beendet ist, ist die Kommandoprozedur in der Bibliothek SYSVOL.MYLIB installiert.

Um einen Kommandoprozessor zu installieren, muß der Prozessor zuerst kompiliert oder assembliert werden. Nachdem der Prozessor mit Hilfe des Link Editors gelinkt ist, muß der Prozessor unter Verwendung des IT (Install task) Kommandos in eine Programmdatei eingetragen werden.

2.4 Verwendung der Kommandos

Wenn die Kommandoprozedur und der Kommandoprozessor installiert sind, kann das Kommando verwendet werden, wenn der SCI ein Kommando anfordert. Falls das Kommando in einer anderen als der Systembibliothek installiert wurde, muß diese Bibliothek dem SCI mitgeteilt werden (unter Verwendung der .USE Anweisung), bevor das Kommando verfügbar ist. Die zuletzt eingegebene .USE Anweisung spezifiziert die aktuelle Kommandobibliothek. Eine .USE Anweisung ohne Angabe eines Pfadnamens spezifiziert die Systembibliothek als aktuelle Kommandobibliothek.

```
[ ] .USE    .MYLIB
[ ] EXP
EXAMPLE PROCEDURE
```

Sobald der SCI aktiviert wird (im interaktiven Modus) arbeitet er wie folgt:

1. Log-in wird gefordert.
2. Die Nachrichtendatei .S$NEWS wird ausgegeben, falls sie existiert.
3. Das Hauptmenü wird angezeigt.
4. Ein Kommando wird angefordert. Das Kommando kann ein Primitivkommando der SCI Sprache sein (siehe Paragraph 4), der Name einer SCI Kommandoprozedur oder der Name eines Menüs. Alle Kommandos, die im Handbuch angegeben sind, sind Namen von Kommandoprozeduren.
5. Falls das Kommando Schlüsselwortparameter besitzt, die noch nicht definiert sind, fordert der SCI den Bediener auf, die Schlüsselwortwerte einzugeben.
6. Falls das Kommando ein Primitivkommando war, führt es der SCI direkt aus. Falls das Kommando ein Prozedurname war, holt der SCI die entsprechende Prozedurdefinition von der aktuellen Kommandobibliothek. Während der Prozessor aktiv ist, wird der SCI suspendiert, es sei den der Prozessor wurde als Hintergrund Task aktiviert.
7. Nachdem der Kommandoprozessor geendet hat, gibt der SCI die terminal lokale Datei des Vordergrundes aus, falls diese nicht leer ist. Falls der Bildschirm Meldungen des Kommandoprozessors anzeigt, kann der Bediener durch Betätigung der CMD-Taste seine Arbeit fortsetzen.
8. Nachdem die terminal lokale Datei angezeigt wurde (oder falls keine angezeigt wurde), kann der SCI Meldungen von anderen Terminals, gesendet durch das CM Kommando (Create Message), oder Status Meldungen von Hintergrund oder Vordergrund Tasks ausgehen. Um fortzusetzen muß der Bediener wie oben beschrieben die CMD-Taste betätigen.
9. Es wird entweder das Menü, das durch das zuletzt ausgeführte Kommando spezifiziert wurde, angezeigt oder das Hauptmenü, falls kein anderes spezifiziert war.
10. Setze bei Schritt 4 fort.

2.5 Löschen eines Kommandos

Um ein Kommando zu löschen, sollte der Anwender die Kommandoprozedur löschen. Falls die Prozedur einen Prozessor aufruft (z.B. durch .BID oder .QBID), der nicht auch von einer anderen Kommandoprozedur aufgerufen wird, sollte dieser ebenfalls aus der Programmdatei gelöscht werden, in der er installiert wurde. Eine Kommandoprozedur kann mit Hilfe des DF Kommandos (Delete File) gelöscht werden. Der PATHNAME der zu löschenden Datei ist:

Kommando Bibliothek Pathname . Kommandoname

Um zum Beispiel die Prozedur EXP zu löschen, die in der Bibliothek SYSVOL.MYLIB installiert ist, ist folgendes Kommando zu verwenden:

DF PATHNAME=SYSVOL.MYLIB.EXP

Um einen Kommandoprozessor zu löschen, ist einfach die Task mit dem DT Kommando (Delete Task) zu entfernen.

3 ENTWURF VON KOMMANDOPROZEDUREN UND PROZESSOREN

Wenn neue Kommandoprozeduren und Prozessoren zu schreiben sind, sollte sich der Anwender gewisser Möglichkeiten, die der SCI bietet und gewisser Restriktionen, die durch die Implementierung vorgegeben sind, bewußt sein. Die folgenden Paragraphen geben allgemeine Hinweise für den Entwurf von Prozeduren und Prozessoren. Außerdem werden die verschiedenen Möglichkeiten und Einschränkungen, die im einzelnen gelten, beschrieben.

3.1 Entwurf von Prozeduren

Im allgemeinen werden Prozeduren geschrieben, um folgende Aufgaben zu erfüllen:

Sammeln der Werte von Schlüsselworten
Aufruf von Prozessoren oder anderer Kommandoprozeduren.
Wertzuweisung an Synonyme.

Einige Prozeduren testen die Werte von Schlüsselworten (unter Verwendung des Primitivkommandos .IF, beschrieben in Paragraph 4.5) und führen in Abhängigkeit von Schlüsselwortwerten verschiedene Aktionen aus. Zum Beispiel prüft das IT Kommando (Install Task) das Schlüsselwort "ATTACHED PROCEDURES?" und ruft in Abhängigkeit vom Ergebnis der Prüfung entweder eine andere Prozedur auf oder weist Synonyme zu.

Die Möglichkeit einer Prozedur andere Prozeduren aufzurufen, eröffnet dem Anwender mehrere Vorteile. So ermöglicht es der einen Prozedur eine einfache Prüfung der Schlüsselworte auszuführen und dann zu einer Anzahl von anderen Prozeduren zu verzweigen. Das CF Kommando (Create File) ist ein Beispiel für eine solche Prozedur.

```
*================================= CF ================================
.PROC  CF (CREATE FILE - SEQ, REL, KEY, DIR, PROG, IMG),
            FILE TYPE = NAME (SEQ)
       .IF  "@$$MO", EQ, 0
            MSG  T="ERROR:  USE SPECIFIC FILE TYPE CREATE IN BATCH"
       .ELSE
            CF$1 "&FILE TYPE" = "*", FORCE=Y
       .ENDIF
.EOP
*================================= CF$1 ==============================
.PROC  CF$1,
            FORCE = YESNO,
            SEQUENTIALFILE = *STRING,
            RELATIVERECORDFILE = *STRING,
            KEYINDEXEDFILE = *STRING,
            DIRECTORYFILE = *STRING,
            PROGRAMFILE = *STRING,
            IMAGEFILE = *STRING
*
       .IF  "&SEQUENTIALFILE", NE, ""
            CFSEQ
            .EXIT
            .ENDIF
       .IF  "&RELATIVERECORDFILE", NE, ""
            CFREL
            .EXIT
            .ENDIF
       .IF  "&KEYINDEXEDFILE", NE, ""
            CFKEY
            .EXIT
            .ENDIF
       .IF  "&DIRECTORYFILE", NE, ""
            CFDIR
            .EXIT
            .ENDIF
       .IF  "&PROGRAMFILE", NE, ""
            CFPRO
            .EXIT
            .ENDIF
       .IF  "&IMAGEFILE", NE, ""
            CFIMG
            .EXIT
            .ENDIF
       MSG  TEXT = "INVALID FILE TYPE"
.EOP
```

Eine Prozedur kann so gestaltet werden, daß sie mehrere Schritte seriell ausführt oder mehrere Prozessoren seriell aufruft. Wenn eine Prozedur im Hintergrund aufgerufen wird (z.B. wenn sie in einer SCI Batchfolge verwendet wird), kann sie andere Prozeduren und Prozessoren entweder im Hintergrund oder im Vordergrund aufrufen, wohin sich jede von diesen fertig melden kann, bevor das nächste Kommando initiiert wird. Wenn jedoch eine Prozedur im Vorgrund arbeitet (z.B. wenn sie von einem Terminal aufgerufen wurde) können mehrere Hintergrundprozesse nicht sequentiell ausgeführt werden. Mehrere Hintergrundprozesse können nur dann sequentiell ausgeführt werden, wenn sie durch die Ausführung einer Batchfolge von SCI Kommandos angestoßen werden.

Wird eine SCI Operation im Batchmodus ausgeführt, muß die Hauptprozedur einer geschachtelten Kommandodefinition alle Schlüsselworte des Kommandos enthalten, einschließlich derjenigen Schlüsselworte, die nur in den inneren Prozeduraufrufen verwendet werden. Zu diesem Zweck kann ein Schlüsselwort in der formalen Schlüsselwortparameterliste ohne Typ oder default value Information (siehe Paragraph 4.1) deklariert werden. Diese Technik ist zu verwenden, wenn eine Kommandoprozedur sowohl im Interaktiv als auch im Batch Modus verwendet werden soll, aber die Schlüsselworte nur im Interaktiv Modus angezeigt werden sollen. Es ist zu beachten, daß ein einzelnes Kommando mehr als 10 Schlüsselworte enthalten kann, aber nur maximal 10 Schlüsselworte am Terminal ausgegeben werden können.

Beispiel:

```
.PROC       RUN (EXECUTE APPLICATION PROGRAM),
                LANGUAGE (COBOL,FORTRAN) = NAME,
!                       == COBOL KEYWORDS:  ==
                OBJECT ACCESS NAME,
                DEBUG MODE?,
!                       == FORTRAN KEYWORDS: ==
                PROGRAM FILE LUNO,
                TASK ID
!                       == CHECK FOR BATCH MODE ==
        .IF  "@$$MO", EQ, 0
            .IF  "&LANGUAGE", EQ, "COBOL"
                XCP    OBJECT ACCESS NAME = "&OBJECT ACCESS NAME",
                       DEBUG MODE? = "&DEBUG MODE"
                .ENDIF
            .IF  "&LANGUAGE", EQ, "FORTRAN"
                XFT    PROGRAM FILE LUNO = "&PROGRAM FILE LUNO",
                       TASK ID = "&TASK ID"
                .ENDIF
            .EXIT
            .ENDIF
!                      == INTERACTIVE MODE ==
        .IF  &LANGUAGE, EQ, "COBOL"
            XCP
            .ENDIF
        .IF  &LANGUAGE, EQ, "FORTRAN"
            XFT
            .ENDIF
.EOP
```

Die Hauptprozedur RUN ist in zwei Abschnitte unterteilt. Der erste Abschnitt ist für den Batchmodus gedacht. Er ruft entweder XCP (execute cobol programm) oder XFT (execute fortran task) mit den entsprechenden Schlüsselworten auf. Der zweite Abschnitt ruft XCP oder XFT ohne Wertzuweisung an die Schlüsselworte auf. Somit werden die betreffenden Schlüsselworte von der aufgerufenen Prozedur selbstständig am Terminal ausgegeben und dadurch der Anwender zur Zuordnung von Werten aufgefordert.

ANMERKUNG

Sobald eine Prozedur beendigt ist, sind ihre Schlüsselworte und deren Werte zerstört. Sie sind nicht mehr in der Tabelle der Schlüsselworte enthalten.

Es ist zu beachten, daß sich Prozeduren auch selbst rekursiv aufrufen können, genauso wie sie andere Prozeduren aufrufen. Diese geschachtelten Prozeduraufrufe bewirken ebenso die Ausgabe der Schlüsselworte, welchen noch kein Wert zugewiesen wurde, falls der SCI im Vordergrund arbeitet (d.h. interaktiv). Beim Schreiben von rekursiven Prozeduren oder bei Prozeduren, die sehr tief geschachtelte Prozeduraufrufe besitzen, muß auf folgendes geachtet werden:

Alle Schlüsselworte und Synonyme einschließlich ihrer Werte, die in einer Prozedur definiert sind, werden in einer Tabelle fester Länge gespeichert, bis die Prozedur beendet ist. Wenn daher eine Prozedur eine andere oder sich selbst aufruft, bleiben alle ihre Schlüsselworte und deren Werte in dieser Tabelle. Die Schlüsselworte der aufgerufenen Prozedur werden in der Tabelle gespeichert. Somit wird die Tabelle aufgefüllt, wenn eine Prozedur rekursiv ist oder eine andere Prozedur aufruft. Wenn die Tabelle aufgefüllt ist, wird das Kommando abgebrochen und die folgende Fehlermeldung auf dem Bildschirm ausgegeben:

KEYWORD TABLE OVERFLOW.

Um das Auffüllen der Tabelle zu vermeiden, sollten die Anwender rekursive Prozeduren, große Schachtelungstiefe und lange Schlüsselworte vermeiden. Eine Möglichkeit das Schachteln zu vermeiden und trotzdem viele Prozeduren aufrufen zu können ist, so viele Prozeduren wie nur möglich auf derselben Stufe aufzurufen. Damit ist gemeint, daß die Folge der Prozeduraufrufe Baumstruktur besitzt (siehe Abb. 3-1); der Baum ist breiter als hoch. Eine Prozeduraufruffolge wie im Baum 1 in Abbildung 3-1 würde bewirken, daß die Schlüsselworte und ihre Werte für die Prozeduren A, B, D, E bis I gleichzeitig in der Tabelle zu halten sind. Eine bessere Aufruffolge wird im Baum 2 gezeigt. Hier muß die Tabelle höchstens die Schlüsselworte für maximal drei Prozeduren gleichzeitig speichern. Natürlich können die Prozeduren, die auf derselben Schachtelungsstufe stehen, nicht auf alle anderen Schlüsselworte zugreifen. Rekursive Prozeduren können gewöhnlich unter Verwendung iterativer Schleifen vermieden werden.

Eine Eigenschaft des SCI, die häufig in Kommando Prozeduren verwendet wird, ist die Möglichkeit Synonyme als Vorgabewerte für Schlüsselworte zu verwenden. Vorgabewerte für Schlüsselworte werden detaillierter in Paragraph 4.1 beschrieben. Ein Vorgabewert für ein Schlüsselwort wird definiert, indem man den Vorgabewert, in Klammern eingeschlossen, hinter die Typdefinition des Schlüsselwortes schreibt.

Zum Beispiel:

```
.PROC      SF(SHOW FILE),
           FILE PATHNAME = *ACNM (@$SF$P)
.SYN          $SF$P = "&FILE PATHNAME"
.IF        "&FILE PATHNAME", NE,   " "
           .SHOW @&FILE PATHNAME
           .ENDIF
.EOP

.PROC      PF(PRINT FILE),
           FILE PATHNAME=ACNM("@$SF$P"),
           ANSI FORMAT?=YESNO(NO),
           LISTING DEVICE=ACNM("@$PF$D")
.OVLY      OVLY=>24,
           PARMS=(@&FILE PHNAME,@&LISTING DEVICE,&ANSI FORMAT?)
.SYN       $PF$D = "&LISTING DEVICE"
.SYN       $SF$P = "&PATHNAME"
.SYN       $$LD = "&LISTING DEVICE"
SOS      DEVICE = "@&LISTING DEVICE"
.EOP
```

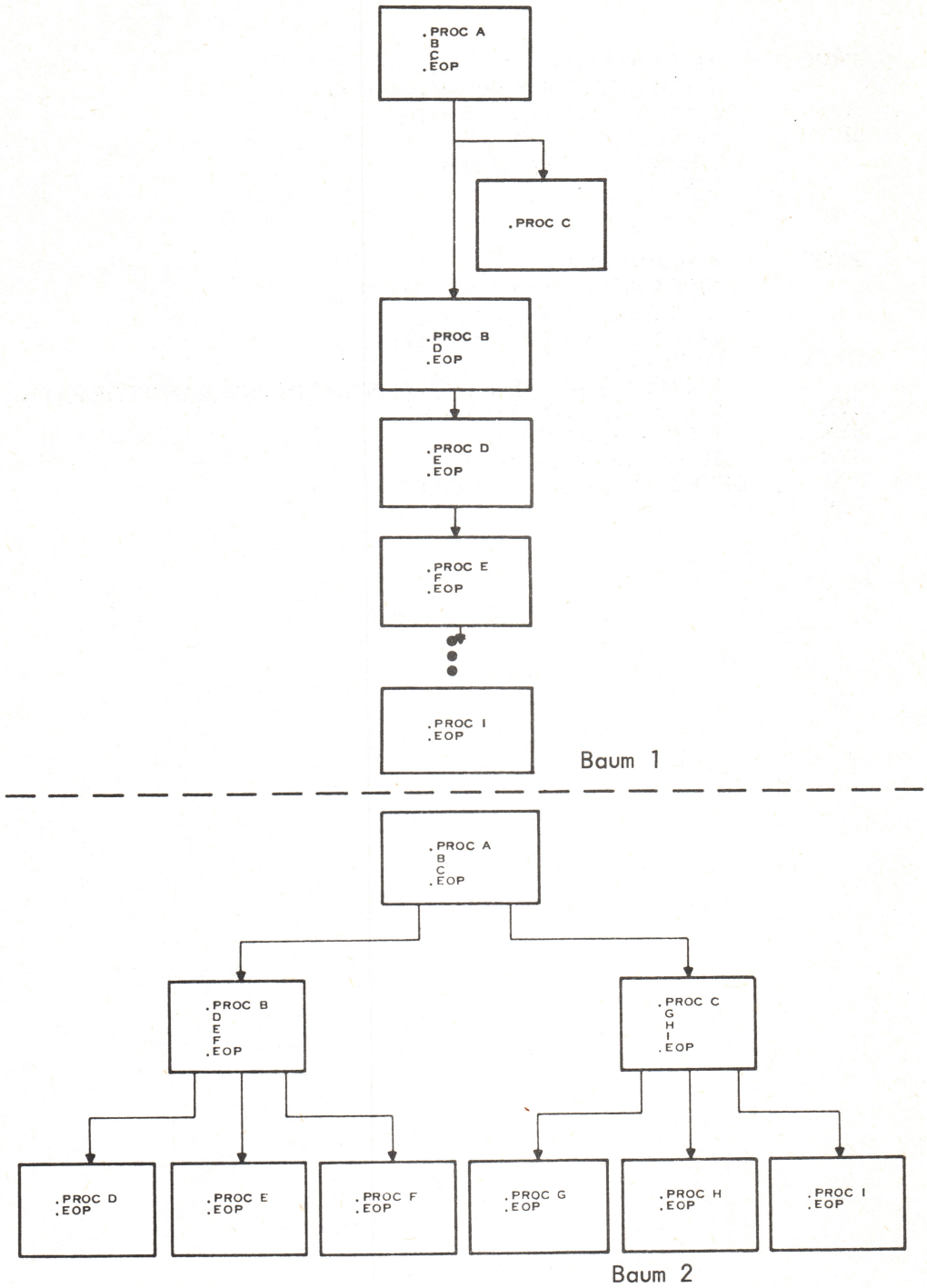


Abb. 3-1: Beispiel für geschachtelte Prozedur-Aufrufe

In der obigen Prozedur ist das Synonym SFP der Vorgabewert für FILE PATHNAME. Das @ Zeichen vor dem Synonymnamen bezeichnet den Wert des Synonyms. Es ist zu beachten, daß SFP nach dem Log-in des Benutzers noch keinen Wert besitzt und daher für das ausgegebene Schlüsselwort noch kein Vorgabewert definiert ist. Nach der ersten Ausführung von SF oder PF ist dem Synonym ein Wert zugewiesen, welcher als Vorgabewert für FILE PATHNAME angezeigt wird. Dieser Vorgabewert gilt dann sowohl für SF als auch PF. Diese Verwendung eines Synonyms als Vorgabewert bewirkt, daß sich eine Prozedur an den letzten Wert, der einem Schlüsselwort zugewiesen wurde, erinnert. Weiter ist anzumerken, daß Vorgabewerte nicht unbedingt Synonyme sein müssen, noch daß Synonyme, die als Vorgabewerte verwendet werden, unbedingt in der Prozedur einen Wert zugewiesen bekommen, in der sie verwendet werden.

ANMERKUNG

Synonymnamen, die mit dem Dollarzeichen ($) beginnen, werden speziell vom SCI verwendet. Falls ein solches Synonym als Vorgabewert für ein Schlüsselwort verwendet wird und das Synonym undefiniert ist, wird der undefinierte Synonymwert (d.h. der Synonymname) vom SCI nicht angezeigt.

Zusätzlich zu den Lokalen Synonymen, die der Anwender erzeugt, werden vom SCI einige globale Synonyme verwaltet, auf die Kommando Prozeduren zugreifen können. Im folgenden werden einige Synonyme aufgezählt, die für eine Prozedur wertvoll sind:

$$MO - Zweistelliger Hexadezimalkode für den SCI Modus
00 = Batch Modus
0F = VDT Modus

S$ST - Zweistellige Dezimalzahl für die Nummer des Terminals (z.B. Ø9)

S$UI - Sechsstellige Zeichenkette für die Anwenderidentifikation (z.B. SYSØØØ)

ME - Vier Zeichen für den Namen des Terminals (z.B. STØ9)

S$CC - Fünfstelliger Hexadezimalkode für die Fertigmeldung. Dieser Kode wird von allen Standard SCI Prozeduren zurückgemeldet.

Schließlich kann durch eine Prozedur der SCI dazu veranlaßt werden ein Menü anzuzeigen oder zu ersetzen. Dies kann mit Hilfe des Primitiv Kommandos .MENU oder das Symbol / geschehen.
Beide sind in Paragraph 4.11 beschrieben.

3.2 Entwurf von Kommando Prozeduren

Für den Entwurf von Kommando Prozessoren gibt es nur wenige allgemein gültige Entwurfskriterien, da Kommando Prozessoren für eine große Anzahl unterschiedlichster Aufgaben geschrieben werden. Ein Prozessor kann entweder in einer höheren Programmiersprache oder in der Assemblersprache geschrieben werden. Ein Prozessor kann auf jedes beliebige Schlüsselwort oder Synonym in der TCA zugreifen. Diese Kommunikation zwischen Prozessor und SCI geschieht mit Hilfe spezieller Schnittstellenroutinen.

Kommando Prozessoren, die eine große Anzahl von Eingabewerten (d.h. mehr als 10-20 Schlüsselworte) anfordern, sollten eine eigene E/A und Parameteranforderung an das Terminal besitzen, statt die gesamte Eingabe durch Schlüsselworte zu realisieren. Außerdem sollten Prozessoren, die eine lange Ausführungszeit besitzen, oder mit Hilfe von langen Eingaben durch Batch Kommandofolgen arbeiten, wie zum Beispiel Compiler oder der Link Editor, im Hintergrund arbeiten, statt im Vordergrund. Um dies zu erreichen, sollten solche Prozessoren von der Kommando Prozedur mit Hilfe des Primitivkommandos .QBID aufgerufen werden. Damit ist das Terminal nicht für weitere Aufgaben blockiert.

4 DIE SYSTEM COMMAND INTERPRETER SPRACHE

Die SCI Sprache besteht aus einer Menge von Kommandos, Primitivkommandos und Sprachelementen, die durch den SCI interpretiert werden. In der folgenden Beschreibung der Sprache und ihrer Syntax, sind die Bezeichner, die in spitze Klammern < > eingeschlossen sind, durch den Anwender zu ersetzen. Bezeichner, die in eckige Klammern [ ] eingeschlossen sind, können entfallen.

Die syntaktische Grundstruktur einer Anweisung der SCI Sprache hat folgende Gestalt:

[Leerzeichen] <Operator> <Leerzeichen> [<Schlüsselwort-Liste>]

mit <Operator> ist ein Kommando oder ein Primitivkommando
Schlüsselwort-Liste ist eine Liste von Schlüsselwortzuweisungen (z.B. K2=INT), die durch Kommas getrennt sind.

Jeder Satz der SCI Sprache, dessen letztes Zeichen ungleich dem Leerzeichen ein "=" oder ein "," ist, wird in der nächsten Zeile fortgesetzt. Das folgende Beispiel wird vom SCI als eine einzige Anweisung aufgefaßt:

```
IDT YEAR=1977,MONTH=4
    DAY=27,HOUR=18,MINUTE=
    56
```

Die SCI Sprache kennt zwei verschiedene Typen von Variablen: Synonyme und Schlüsselworte. Synonymwerte werden mit Hilfe des Primitivkommandos .SYN (siehe Paragraph 4.2) definiert und sind dem Anwender zugeordnet, der sie definiert hat. Eine Bezugnahme auf den Wert eines Synonyms wird durch den Synonymnamen dargestellt, welchem das @ Zeichen vorangestellt ist. @XYZ bezieht sich auf den Wert des Synonyms XYZ. Der Wert eines undefinierten Synonyms ist der Synonymname selbst. Immer, wenn ein Synonymname, dem ein @ vorausgeht, in einer Zeichenkette oder einem Ausdruck auftaucht, ersetzt der Synonymwert das Synonym.
Zum Beispiel wird die Zeichenkette

"LISTING DEVICE IS @DEVICE"

durch den SCI als

"LISTING DEVICE IS LPØI"

interpretiert, falls DEVICE vorher mit LPØI definiert wurde. Falls DEVICE undefiniert ist (d.h. kein Wert zugewiesen wurde), wird die Zeichenkette mit

"LISTING DEVICE IS DEVICE"

interpretiert.

Die Auswertung von Synonymen (angestoßen durch die Verwendung von @) wird wie folgt durchgeführt: Wenn der SCI auf das Zeichen @ stößt, vergleicht er den sich daran anschließenden Teil der Zeichenkette, beginnend von @ bis zum ersten Sonderzeichen (d.h. dem ersten Zeichen, das kein Großbuchstabe des ASCII Kodes ist), mit den Namen in der Synonymtabelle. Falls diese Teilzeichenkette mit keinem Namen aus der Symboltabelle identisch ist, gilt das Synonym als undefiniert und es wird ihm als Wert die Zeichenkette selbst zugewiesen.

Wegen der Bedeutung der Sonderzeichen bei der Auswertung von Synonymen, kann die Verwendung von Synonymen als Platzhalter von Pfadnamen Probleme ergeben. Falls ein Synonym verwendet wird, um einen ganzen Pfadnamen oder dessen ersten Teil zu ersetzen, gibt es keine Probleme. Anders ist es, wenn ein Synonym die zweite Komponente eines Pfadnamens vertritt (z.B. in VOL1.MYCAT.S sei S ein Synonym mit dem Wert SOURCE); das Synonym wird nicht richtig ausgewertet, wenn vor den Pfadnamen ein @ geschrieben wird (d.h. "@VOL1.MYCAT.S"). Synonyme können dann als zweite Komponenten eines Pfadnamens verwendet werden, wenn die @ Zeichen richtig im Sinne der Auswertung der Zeichenkette verwendet werden (z.B. "@VOL1.@MYCAT.@S" wird ausgewertet als VOL1.MYCAT.SOURCE).

Ein Schlüsselwort wird in einer .PROC Schlüsselwortliste (siehe Paragraph 4.1) als Zeichenkette links vom Gleichheitszeichen "=" definiert. In der Anweisung (.PROC LP,FILE=ACNM) ist FILE ein Schlüsselwort. Einem Schlüsselwort wird dann ein Wert zugewiesen, wenn die zugehörige Kommando Prozedur aufgerufen wird. Der leere Wert eines Schlüsselwortes ist die leere Zeichenkette (d.h. Zeichenkette der Länge Null). Der Bezug auf den Wert eines Schlüsselwortes wird durch den Namen des Schlüsselwortes dargestellt, dem das &-Zeichen vorausgeht. Zum Beispiel bezieht sich &INPUT FILE PATHNAME OR LUNO auf den Wert des Schlüsselwortes INPUT FILE PATHNAME OR LUNO. Es ist zu beachten, daß die Leerzeichen innerhalb eines Schlüsselwortes von Bedeutung sind. Die maximale Schlüsselwortlänge beträgt 28 Zeichen, einschließlich der im Schlüsselwort eingebetteten Leerzeichen.

Manchmal wird einem Schlüsselwort als Wert der Name eines Synonyms zugewiesen. Um nun eine richtige Auswertung eines solchen Schlüsselwortes zu erreichen, sollte die Konstruktion

"@& <Schlüsselwort> "

verwendet werden. & <Schlüsselwort> wird durch den Schlüsselwortwert <Schlüsselwert> ersetzt. Falls <Schlüsselwert> ein Synonymname ist, ergibt die entgültige Auswertung den Wert des Synonym <syn-Wert> . Falls <Schlüsselwert> kein Synonym ist, wird @<Schlüsselwert> als das leere Synonym ausgewertet. Da aber das leere Synonym (d.h. undefinierte Synonym) als Wert den Synonymnamen besitzt, ergibt sich als Ergebnis der Auswertung <Schlüsselwert> als Wert.

Schlüsselworten und Synonymen können Zeichenketten als Werte zugewiesen werden, die Zeichen enthalten, welche für den SCI eine besondere Bedeutung besitzen (z.B. Komma, Gleichheitszeichen, Klammern). Wenn solche Variable in einer Zeichenkette unter Verwendung der Zeichen "&" und "@" ausgewertet werden sollen, sollten solche Variable in Hochkommas (") geschrieben werden, um Mißverständnisse zu vermeiden.

Da Schlüsselworte manchmal recht lang sind, erlaubt der SCI diese Schlüsselworte abzukürzen, wenn sie in einer Wertzuweisung auftreten. Diese Methode ist besonders dann nützlich, wenn die Schlüsselworte im Batch Modus auftreten, weniger im interaktiven Modus. Die Kurzform der Schlüsselworte wird entsprechend der folgenden Regeln bestimmt:

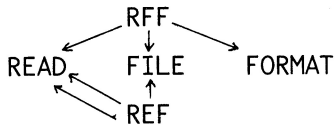
A sei die Langform eines Schlüsselwortes, und B seine Abkürzung.

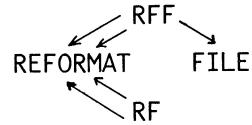
1. A und B müssen Schlüsselworte derselben Kommando Prozedur auf derselben Schachtelungsstufe sein.
2. A und B müssen mit demselben Zeichen beginnen.
3. Alle Zeichen in B müssen auch in A vorkommen und dieselbe Reihenfolge wie in A besitzen.
4. Leerzeichen in A haben die folgende Eigenschaft:
   Das erste Zeichen in dem Teil von B das mit A nicht übereinstimmt, muß mit dem Zeichen übereinstimmen, das dem Leerzeichen in A folgt.
5. Dezimalzahlen in A müssen vollständig in B auftreten und in derselben Reihenfolge wie in A.
6. Zeichen, die in A nach einem Sonderzeichen stehen (nicht alphanumerisch oder Leerzeichen), werden ignoriert.

Beispiel: Die Langform eines Schlüsselwortes sei INPUT FILEØ3 (PATHNAME)

Abkürzung	Übereinstimmung?	Verletzte Regel (n)
FØ3	NEIN	2
IFØ3	JA	
ITPØ3	NEIN	3,4
ITFØ3	JA	
IFLØ3	JA	
IPL	NEIN	4,5
IFØ3P	NEIN	6
IFP	NEIN	5,6

Besonders sorgfältig sollte darauf geachtet werden, daß die Abkürzung mit dem richtigen Schlüsselwort übereinstimmt. Wenn zum Beispiel READ FILE FORMAT und REFORMAT FILE? zwei Schlüsselworte eines Kommandos sind, stimmen die Abkürzungen REF, RF oder RFF sowohl mit dem einen als auch mit dem anderen Schlüsselwort überein.





Wenn ein Schlüsselwort in der .PROC Anweisung definiert wird, ist dieser Variablen eine Typerklärung zuzuweisen. Die gültigen Typen für Schlüsselworte sind in Tabelle 4-1 angegeben:

Tabelle 4-1

Eingabetyp	Definition
STRING	Eine Zeichenkette (eine Kette von Zeichen, die kein Doppelausführungszeichen ("), Ausrufezeichen, Gleichheitszeichen, Klammern oder Kommas enthalten) oder eine Kette eingeschlossen in Doppelausführungszeichen (eine Kette von Zeichen, die selbst kein Doppelausführungszeichen enthält, aber in solche eingeschlossen ist).
ACNM	Ein Pfadname für eine Datei oder ein Gerätename (ACNM, ACCESS NAME).
INT	Ein hexadezimaler oder dezimaler ganzzahliger Ausdruck (INT, INTEGER). Eine führende Null oder ein ">" definieren eine Hexadezimalkonstante.
YESNO	Eine alphabetische Zeichenkette, die mit Y oder N beginnt.
NAME	Eine Zeichenkette, die nur Buchstaben und Ziffern enthält und mit einem Buchstaben (A-Z) beginnt.

Der Wert eines Schlüsselwortes kann ein einziger Wert oder eine Werteliste sein. Eine Werteliste kann für jedes Schlüsselwort definiert werden, indem man die Typerklärung in Klammern schreibt. Zum Beispiel definiert FILE=ACNM das Schlüsselwort FILE vom Typ ACNM. Die Definition INPUT=(ACNM) gibt an, daß das Schlüsselwort INPUT entweder einen einzigen Wert oder eine Werteliste vom Typ ACNM bezeichnen kann. Eine List wird als eine Folge von einzelnen Angaben geschrieben, welche durch Kommas getrennt sind und in Klammern eingeschlossen ist. Damit sind FILE=DSØ4.LIST und INPUT=(MY.INPUT.F1,MY.INPUT.F2) gültige Zuweisungen für FILE und INPUT. Falls eine Liste nur ein einziges Element enthält, sind die Klammern nicht nötig.

Die Konkatenation wird dadurch angegeben, daß Zeichenketten, Ketten und Variable unmittelbar hintereinander geschrieben werden. Damit konkateniert ABC&FILE die Zeichenkette ABC mit dem Wert des Schlüsselwortes FILE, während LAST @XYZ die Zeichenkette LAST, ein Leerzeichen und den Wert des Synonyms XYZ konkateniert.

Das Zeichen "!" kennzeichnet das Ende eines Satzes, wenn es in einer SCI Anweisung verwendet wird. Kommentare können an eine SCI Anweisung angefügt werden, wenn sie mit dem Ausrufungszeichen beginnen. Kommentare können jedoch nicht über einen Satz (Zeile) fortgesetzt werden. Zusätzlich wird eine ganze Zeile als Kommentaranweisung interpretiert, wenn die Zeile in der ersten Spalte einen Stern (*) enthält.

In Tabelle 4-2 sind alle SCI Primitiv Kommandos und ihre Parameter aufgeführt.

Tabelle 4-2 SCI Primitiv Kommandos.

.PROC	<name> [(<voller Name>)] [=<int>,] [<formale Schlüsselwort Liste>]
.EOP	
.DATA	<acnm>
.EOD	
.SYN	<name>="<string>"
.EVAL	<name>=<ganzzahliger Ausdruck>
.BID	TASK =<int>[,LUNO =<int>] [, CODE =<int>] [, PARMS = (<string>)]
.QBID	TASK =<int>[,LUNO =<int>] [, CODE =<int>] [, PARMS = (<string>)]
.DBID	TASK =<int>[,LUNO =<int>] [, CODE =<int>] [, PARMS = (<string>)]
.SPLIT	LIST = <string>,FIRST =<name>[,REST = <name>]
.IF	"<string>",<Relation>, "<string>"
.ELSE	
.ENDIF	
.LOOP	
.UNTIL	"<string>",<Relation>, "<string>"
.WHILE	"<string>",<Relation>, "<string>"
.REPEAT	
.EXIT	
.USE	<acnm>
.MENU	<menu name>
.STOP	
.SHOW	<acnm>
MSG	TEXT =<string>[,REPLY =<name>]

4.1 Primitiv Kommando für die Prozedur Definition

Eine SCI Prozedur beginnt mit der .PROC Anweisung und endet mit einer .EOP Anweisung. Das .PROC Primitiv Kommando hat die folgende Form:

.PROC <name> [(<voller Name>)] [= <integer>] [, < Schlüsselwortliste>]

<name> muß unmittelbar hinter .PROC stehen und kann nicht entfallen. Eingeschlossen in runde Klammern kann <voller Name> angegeben werden. Er muß, falls angegeben unmittelbar nach <name> stehen. < Voller name > wird auf dem Bildschirm ausgegeben, wenn die Prozedur angewählt wird. Das wahlfreie Feld = <integer> muß, falls angegeben, unmittelbar hinter < voller Name> stehen und bestimmt die Privilegstufe des Kommandos. Die Tabelle 4-3 zeigt die verschiedenen Privilegstufen. Die Privilegstufen von Kommandos haben die folgende Bedeutung: einem Anwender, der eine Privilegstufe besitzt, die kleiner ist als die, die einer bestimmten Kommando Prozedur zugeordnet ist, wird die Verwendung des betreffenden Kommandos verweigert. Sonst können einerseits die SCI Kommandos entsprechend ihrer Mächtigkeit und/oder Gefährlichkeit und andererseits die Privilegstufen der Anwender entsprechend ihrer Vertrauenswürdigkeit und/oder ihres Kenntnisstandes hierarchisch gegliedert werde

Tabelle 4-3 Privilegstufen von Kommandos

Stufe	Bedeutung
0	Vom Benutzer definiert (Niedrigste Stufe des Zugriffsprivilegs)
1	Vom Benutzer definiert
2	Vom Benutzer definiert
3	Vom Benutzer definiert
4	Zugriffsprivileg für allgemeine Systemdienste
5	Zugriffsprivileg für Verwaltungsdienste
6	Kombination von System- und Verwaltungsdiensten
7	reserviert

Eine <Schlüsselwortliste> ist von der Form < Schlüsselwort> = < Eingabetyp> , wobei <Schlüsselwort> eine Zeichenkette ist (Name eines Schlüsselwortes) und < Eingabetyp> einer der gültigen Typen, die in Tabelle 2-1 zusammengefaßt sind. Der SCI prüft die Richtigkeit der Eingabewerte, wenn die Prozedur aufgerufen wird. Wahlweise kann die Form für ein Schlüsselwort wie folgt angegeben werden: < Schlüsselwort> = <Eingabetyp> (< Vorgabewert >), um ein Schlüsselwort wird einem Vorgabewert vorzubesetzen. < Vorgabewert > kann eine Zeichenkette, eine Kette oder eine Variable (aber keine Liste) sein. Steht vor dem Eingabetyp ein Stern "*", bedeutet dies, daß das Schlüsselwort entfallen kann.
Beispiel: FILE=*ACNM bedeutet, daß FILE ein wahlfreies Schlüsselwort ist. Schlüsselworte, die nicht wahlfrei sind, müssen definiert sein, bevor der SCI die Prozedur ausführt, während wahlfreie Schlüsselworte nicht definiert sein müssen.

.EOP beendet die Definition einer Kommando Prozedur

Beispiel:

```
.PROC EX (EXAMPLE PROC) =0,
        INPUT PATHNAME = *NAME,
          DISPLAY (YES/NO)?=*YESNO(NO),
        OUTPUT PATHNAME = NAME
  SF      FILE = "&INPUT PATHNAME"
  .EOP
```

4.2 Primitiv Kommandos für die Synonymbehandlung

Es gibt drei Primitivoperationen (.SYN,.EVAL,.SPLIT) mit deren Hilfe man einem Synonym Werte zuweist.

.SYN <name> =" <wert> "

Der .SYN Operator weist einem Synonym einen Wert zu. <name> ist der Name eines Synonyms. <wert> kann eine Kette, eine Zeichenkette, eine Variable oder ein konkatenierter Ausdruck sein. Die einschließenden Doppelhochkommas sind äußerst wichtig, um den Wert richtig zu interpretieren. Wird einem Synonym der leere Wert ("") zugewiesen, wird das Synonym aus der NCT (name correspondence table) gelöscht. Synonyme, die nicht mehr benötigt werden, sollten grundsätzlich gelöscht werden, damit die NCT nicht unnötig gefüllt wird.

.EVAL <name> =" <wert> "

.EVAL wertet die <wert> Kette als numerischen Ausdruck aus, konvertiert das Ergebnis in einen dezimalen ASCII Wert und speichert diesen als Wert des Synonyms <name> . Zum Beispiel weist

.EVAL RESULT="@THREE*5-@TWO"

dem Synonym RESULT den Wert "13" zu (vorausgesetzt die Synonyme THREE und TWO haben die entsprechenden Werte "3" und "2").

.SPLIT LIST= <liste> [,FIRST <name1>] [,REST= <name2>]

Der .SPLIT Operator dient dazu den ersten Ausdruck aus einer Werteliste zu entfernen. <liste> ist ein Kettenausdruck, dessen Werte eine Werteliste oder einzelne Werte sein können. Der erste Ausdruck der Liste wird als Wert dem Synonym <name1> zugeordnet, während der Rest der Liste dem Synonym <name2> als Wert zugewiesen wird. Die folgende Tabelle zeigt die Wirkungsweise von .SPLIT:

LIST	FIRST	REST
(A,B,C)	A	(B,C)
A	A	leer
leer	leer	leer
((X,Y),Z,G)	(X,Y)	(Z,G)

Beispiel:

```
.PROC EXP(EXAMPLE PROC) =0,
        INPUT PATHNAME = *NAME (@$EX$IP),
            DISPLAY (YES/NO)? = *YESNO (NO),
        OUTPUT PATHNAME(S) = (NAME)
    .SYN $EX$IP = "&INPUT PATHNAME"          !"REMEMBER" INPUT PATHNAME
    .SYN EXFILES = "(&OUTPUT PATHNAME)"
    .SPLIT LIST = "@EXFILES"                 ! GET THE FIRST
           FIRST = EX$OP,                    !     FILE TO BE WRITTEN
           REST = EXFILES
    .SF    FILE = @@$EX$IP                   ! SHOW THE INPUT FILE
    .EOP
```

4.3 Das Primitiv Kommando DATA

Unter Verwendung des .DATA Primitiv Kommandos können Daten direkt von einer Eingabe Batch Kommandofolge in eine Datei einkopiert werden.

.DATA <pathname>

leitet den Datenstrom ein, während das Primitiv Kommando

.EOD

ihn abschließt. Die Daten werden in diese Datei einkopiert, die mit <pathname> spezifiziert ist.

4.4 Primitiv Kommandos zum Aufruf von Kommando Prozessoren

Die drei Primitiv Kommandos .BID,.QBID und .DBID dienen dazu Kommando Prozessoren (TAXO Tasks) aufzurufen. Jedes Primitiv Kommando besitzt Operanden, die den Kommando Prozess identifizieren und an ihn Parameter übergeben.

.BID TASK= <int> [,LUNO= <int>] [,CODE= <int>] [,PARMS=(<string>)]

.BID bewirkt die Ausführung einer TAXO Task als Vordergrund Task eines Terminals. <int> muß eine Ganzzahl sein, die eine Task in einer Programmdatei spezifiziert. Der Programmdatei muß vorher eine LUNO zugewiesen sein. LUNO, CODE und PARMS sind wahlfreie Parameter. Der Standardwert für LUNO bezeichnet die System Programmdatei. CODE ist eine Ganzzahl (0...255), auf die die Task als Binärwert zugreifen kann. PARMS ist eine Liste von Zeichenketten, die durch Kommas getrennt sind. Auf diese Zeichenketten kann die Task zugreifen. Der SCI lädt die Task und übergibt an sie durch .BID die Kontrolle. Wenn die Task beendet ist, führt der SCI die nächste Anweisung aus.

.QBID TASK= <int> [,LUNO= <int>] [,CODE= <int>] [,PARMS=(<string>)]

.QBID bewirkt die Ausführung einer TAXO Task als Hintergrund Task eines Terminals. Im interaktiven Modus führt der SCI das nächste Kommando aus, nachdem die Ausführung der Task angestoßen wurde. Im Batch Modus wird der SCI solange suspendiert bis die Task beendet wurde. Die Operanden für .QBID sind identisch zu .BID.

.DBID TASK= <int> [,LUNO= <int>] [,CODE= <int>] [,PARMS=(<string>)]

.DBID arbeitet wie das .QBID Primitiv Kommando mit dem Unterschied, daß die Hintergrund Task suspendiert bleibt. Der Sinn dieser Vorgehensweise ist der, daß der Anwender nun den Kommando Prozessor mit Hilfe des Testsystems "Debugger" testen kann.

4.5 Bedingte Primitiv Kommandos

Das bedingte Primitiv Kommando der SCI Sprache entspricht der üblichen IF-THEN bzw. IF-THEN-ELSE Konstruktion und hat folgende Form:

```
.IF     " <op1> ", <Relation> ," <op2> "
    :
.ELSE
    :
.ENDIF
```

wobei <op1> und <op2> Ketten, Variable oder Konkatenierte Ketten sein können. <Relation> ist eine Vergleichsoperation für eine Kette oder ein numerisches Element. Die folgende Tabelle zeigt alle Relationen und ihre Bedeutung:

Relation	Bedeutung
EQ	<op1> ist identisch mit <op2>
NE	<op2> ist nicht identisch mit <op2>
GT	<op1> folgt <op2> in der ASCII Sortierreihenfolge
LT	<op1> steht vor <op2> in der ASCII Sortierreihenfolge
GE	entweder GT oder EQ ist wahr
LE	entweder LT oder EQ ist wahr

Falls sowohl <op1> als auch <op2> als ganzzahlige Konstanten oder Ausdrücke betrachtet werden können, wird der Vergleich als vorzeichengerechter arithmetischer Vergleich durchgeführt. Andernfalls werden beide Operanden als ASCII Zeichenketten betrachtet und entsprechend verglichen.

Zwischen .IF und .ENDIF können beliebige andere SCI Primitiv Kommandos oder Aufrufe für andere Prozeduren stehen, einschließlich anderer .IF Anweisungen. Geschachtelte bedingte Anweisungen dürfen eine Tiefe von 32 Schachtelungen nicht übersteigen.

Das folgende Beispiel ist eine Erweiterung der obigen Beispielprozedur unter Verwendung der Primitivkommandos für die Synonymbehandlung. Außerdem wird das .IF Primitiv Kommando verwendet, um den rekursiven Aufruf des CC Kommandos zu erzeugen.

```
.PROC   EXP(EXAMPLE PROC) = 0,
            INPUT PATHNAME = *NAME (@$EX$IP),
                DISPLAY (YES/NO)? = *YESNO (NO),
            OUTPUT PATHNAME(S) = (NAME)
        .SYN $EX$IP = "&INPUT PATHNAME"      ! "REMEMBER" INPUT PATHNAME
        .SYN EXFILES = "(&OUTPUT PATHNAME)"
        .SPLIT LIST = "@EXFILES",            ! GET THE FIRST
                FIRST = EX$OP,               !    FILE TO BE WRITTEN
                REST = EXFILES
        .IF "&DISPLAY", GE, Y
                SF FILE = @@$EX$IP                ! SHOW THE INPUT FILE
                .ENDIF
        CC      OP = "@@EX$OP", IP = "@@$EX$IP"
        .IF "@EXFILES", NE, " "
                EXP OP = "@EXFILES", D = "&DISPLAY"
                .ENDIF
.EOP
```

4.6 Verlasse eine Prozedur

.EXIT

Dieses Primitiv Kommando beendet die Ausführung (nicht die Definition; dies geschieht durch .EOP) einer Kommando Prozedur. Dieses Kommando kann beliebig oft und an beliebigen Stellen in einer Kommando Prozedur auftreten. Das .EOP Primitiv Kommando am Ende einer Prozedur beendet automatisch deren Ausführung.

```
.PROC       NP (NEW PROC),
        .IF                     ! IF THE CONDITION IS
            CPROC               !   TRUE, EXECUTE
            DPROC               !   THESE TWO COMMANDS
            .EXIT               !   AND QUIT
        .ENDIF
        EPROC                   ! OTHERWISE DO THIS
        .EOP
```

Bei obigem Beispiel ist zu beachten, daß dasselbe Ergebnis auch unter Verwendung eines .ELSE Zweiges erreicht wird.

```
.IF
        CPROC
        DPROC
  .ELSE
        EPROC
  .ENDIF
```

4.7 Verwendung einer alternativen Prozedur Bibliothek

.USE <pathname>

Durch das .USE Primitiv Kommando wird dem SCI die Verwendung einer anderen Prozedurbibliothek mitgeteilt. Die Bibliothek ist in dem Directory zu finden, das durch <pathname> spezifiziert ist. Alle Kommandos, welche nach einer .USE Anweisung aufgerufen werden, werden von dieser Bibliothek genommen. Mehrere .USE Anweisungen hintereinander sind zulässig. Es wird jedoch nur die letzte Anweisung zur Auswirkung kommen. Um wieder zur Standard System Bibliothek (.S$PROC) umzuschalten, ist nur ein .USE Kommando ohne Operand nötig. Wenn eine .PROC Anweisung verwendet wird, muß zuerst ein .USE <pathname> geschrieben werden, damit die neue Kommando Prozedur in die so definierte Bibliothek eingetragen wird.

```
.USE    .MYFILES.PROCS      !SCHREIBE IN DIESE BIBLIOTHEK
.PROC   NP (NEW PROC)
  :
.EOP
```

ANMERKUNG

Wenn mit Hilfe der .USE Anweisung die Standard System Bibliothek durch eine andere Bibliothek ausgetauscht wird, wird dadurch nicht die Bibliothek ausgetauscht, von der die Menüs gelesen werden. Menüs werden immer aus der Standard Bibliothek .S$PROC gelesen.

4.8 Zeige eine Datei

.SHOW <pathname>

Das .SHOW Kommando gibt den Inhalt einer spezifizierten Datei aus. Für interaktive Terminals wird dieser Inhalt auf dem Bildschirm angezeigt; bei Batchverarbeitung, wird der Inhalt auf den beim XB (Execute Batch) Kommando spezifizierten LISTING ACCESS NAME ausgegeben. Das .SHOW Primitiv Kommando arbeitet wie das Standard SCI Kommando SF (SHOW FILE).

4.9 Gib eine Meldung aus

Dieses Primitiv Kommando ist ein Pseudokommando, das wie eine Prozedur implementiert ist und daher nicht mit einem Punkt "." beginnt.

MSG TEXT= <string> [,REPLY= <name>]

MSG bewirkt die Ausgabe einer Meldung am .Terminal, falls der SCI im Vordergrund arbeitet. Falls REPLY angegeben wird, wartet der SCI auf eine Antwort durch den Bediener. Diese Antwort wird als Wert dem Synonym <name> zugewiesen. MSG wird im Batch Modus ignoriert.

Beispiel:

```
.PROC      NK (NO KEYWORDS)
MSG TEXT="KEYWORD",REPLY=SYN
MSG TEXT="KEYWORD2",REPLY=SYN2
.IF  "@SYN",NE,""
                    MSG TEXT="SUCCESSFUL"
.ENDIF
.EOP
```

4.10 Beende die Arbeit des SCI

.STOP

Das .STOP Kommando beendet die Ausführung des SCI.

4.11 Zeige ein Menü

.MENU <menüname>

Die Verwendung des .MENU Primitiv Kommandos bewirkt, daß der SCI das spezifzierte Menü bei nächsten Menüanzeigezyklus auf dem Bildschirm anzeigt (d.h. bevor das nächste Kommando vom SCI angefordert wird). Es

gibt drei Varianten von <menüname> Parametern:

1. kein Menü spezifiziert - die Anwendung von .MENU ohne die Angabe eines Menünamens bewirkt, daß der SCI im nächsten Menüanzeigezyklus kein Menü ausgibt (der Bildschirminhalt bleibt erhalten).
2. <menüname> - falls ein Menüname (1-6 alphanumerische Zeichen) angegeben ist, wird der SCI das Menü im nächsten Menüanzeigezyklus ausgeben.
3. * <menüname> - siehe 2.

Es ist anzumerken, daß das "/" Symbol genau auf dieselbe Weise arbeitet, wie .MENU:

```
/      = .MENU
/DEV   = .MENU.DEV
/*DEV  = .MENU*DEV
```

4.12 Iterative Schleifen

Die folgenden Primitiv Kommandos bewirken die iterative Wiederholung von SCI Anweisungsblöcken:

```
.LOOP
.UNTIL   " <op1> ", <Relation> ," <op2> "
.WHILE   " <op1> ", <Relation> ," <op2> "
.REPEAT
```

Die Parameter <op1> , <Relation> und <op2> haben dieselbe Bedeutung wie in der .IF Anweisung. Die Grundstruktur einer Schleife in einer SCI Prozedur ist wie folgt:

```
.LOOP
  :      SCI Anweisungen
  :
.UNTIL   oder .WHILE
  :      SCI Anweisungen
  :
.REPEAT
  :
  :
```

Eine Schleife wird durch eine .LOOP Anweisung begonnen und mit einer .REPEAT Anweisung beendet. Die Schleife muß mindestens eine .WHILE oder .UNTIL Anweisung enthalten, welche an beliebiger Stelle in der Schleife stehen können und mehrfach auftreten dürfen. Die SCI Anweisungen innerhalb einer Schleife werden fortlaufend ausgeführt bis die Bedingung in einer .WHILE Anweisung den Wert 'FALSCH', oder die Bedingung in einer .UNTIL Anweisung den Wert 'WAHR' erhält. Anschließend führt der SCI die erste Anweisung aus, die dem .REPEAT folgt.

ANMERKUNG

SCI Schleifen können ebenso wie .IF Anweisungen geschachtelt sein. Die maximale Schachtelungstiefe beträgt 32. Insgesamt darf die Schachtelungstiefe von Schleifen und IF Anweisungen zusammen 32 nicht übersteigen.

Das folgende Beispiel zeigt die Anwendung der Schleifenanweisungen. Es wird hier dieselbe Funktion realisiert, wie im Beispiel für die .IF Anweisung.

```
.PROC     EXP (EXAMPLE PROC) = 0
              INPUT PATHNAME = *NAME (@$EX$IP),
                  DISPLAY (YES/NO)? = *YESNO (N),
              OUTPUT PATHNAME(S) = (NAME)
          .SYN $EX$IP = "@&INPUT PATHNAME"        ! REMEMBER PATHNAME
          .SYN EXFILES = "(&OUTPUT PATHNAME)"
          .IF "&DISPLAY", GE, Y
                SF FILE = @@$EX$IP
          .LOOP
          .SPLIT LIST = "@EXFILES",               ! GET THE FIRST
                FIRST = EX$OP                     !    FILE TO BE
                REST = EXFILES                    !    WRITTEN
          CC OP = "@@EX$OP", IP = "@@$EX$IP"
          .UNTIL "@EXFILES", EQ, "EXFILES"
          .REPEAT
.EOP
```

5 MENÜS

Wie in Paragraph 1.3 beschrieben, sind Menüs unter SCI sequentielle Dateien mit 80 Zeichen pro Satz und maximal 23 Sätzen pro Menü. Wenn der SCI ein Menü anzeigt, wird einfach die Menüdatei angezeigt. Alle Menüs sind im .S$PROC Directory auf der Systemplatte enthalten und dort in Dateien mit den Namen M$ <name> aufgeschrieben. Zum Beispiel ist das DEV Menü (DEVice operations) in der Datei .S$PROC.M$DEV enthalten.

Um ein Menü hinzuzufügen oder zu ändern, muß der Anwender nur mit Hilfe des Text Editors das Menü editieren und entweder ein schon bestehendes Menü ersetzen oder ein neues hinzufügen. Neue Menüs müssen im .S$PROC Directory als M$ Dateien angelegt werden.

ANMERKUNG

Unabhängig davon, ob eine andere Kommandobibliothek durch eine .USE Anweisung spezifiziert wurde, bildet der SCI ausschließlich Menüs der .S$PROC Bibliothek ab.

Die Menüdateien sollten die Namen und Beschreibungen anderer Menüs und/oder Kommandos enthalten, ähnlich wie dies in den SCI Standardmenüs geschieht.

Die Veränderung des Hauptmenüs (dasjenige Menü, das automatisch angezeigt wird, wenn der SCI aufgerufen wird) ist ein etwas komplizierter Vorgang. Der Hauptmenüdatei .S$PROC.M$LC ist eine LUNO zugewiesen und solange schreibgeschützt, solange sie eröffnet ist. Damit kann diese Datei solange nicht überschrieben werden, solange der SCI an einem beliebigen Terminal im VDT Modus arbeitet.

Um das Hauptmenü zu ändern, muß wie folgt vorgegangen werden:

1. Erzeuge das neue Menü in einer Scratch Datei.
2. Ändere den Status eines Terminals in den TTY Modus.
3. Beende die Arbeit an allen Terminals (Log-off), welche im VDT Modus arbeiten.
4. Rufe den SCI an dem TTY Terminal.
5. Kopiere das neue Menü in die Datei .S$PROC.M$LC.

Die vom Anwender erzeugte .M$LC Datei enthält nun das Hauptmenü.

TA Computer Systeme

Triumph – Adler Vertriebs-GmbH
Abteilung Systemplanung
Fürther Straße 212
D 8500 Nürnberg
Telefon 0911/322-0

Alle Rechte, sowie Änderungen und Verbesserungen behalten wir uns ohne Ankündigung vor.

All rights reserved, including the right to make alterations and improvements without previous notice.

Nous nous réservons tous les droits ainsi que l'application de toute modification ou amélioration, sans avis préliminaire.

Nos reservamos todos los derechos así como hacer modificaciones y mejoras sin previo aviso.

Ci riserviamo tutti i diritti, come pure modifiche e correzioni senza pre - avviso.